

EXAMEN SIMULADO - ANALISIS DE SISTEMAS

Royer Mendez Ramirez
A43333

June 7, 2020

1 Introducción

Este proyecto trata de la simulación de las salidas del biosistema presentado en el enunciado del presente trabajo. Se presentaran las gráficas de las simulaciones obtenidas y a su vez se hará una reseña de los pasos seguidos para lograr este objetivo. Se obtuvieron las simulaciones por medio de la función S-Function de Simulink y la función ODE45 de Matlab.

2 Modelo S-Function de Simulink

2.1 Análisis y Resultados

S-Function es una herramienta interesante porque esta emplea un método mas gráfico, por medio de bloques que serán programados por medio de código.

El método empleado fue el de hacer un modelo con tres entradas, un bloque de estados y dos salidas que corresponden a X y a Sorg.

Este modelo se simuló dos veces, una vez para ver la gráfica de la simulación con la variación de las salidas al aplicársele una variación de +0.2 a la entrada Sorgo y la otra al aplicársele una variación de -0.2 en la misma entrada.

La figura 1 muestra la gráfica obtenida de la simulación al aplicársele la variación de +0.2 a dicha entrada.

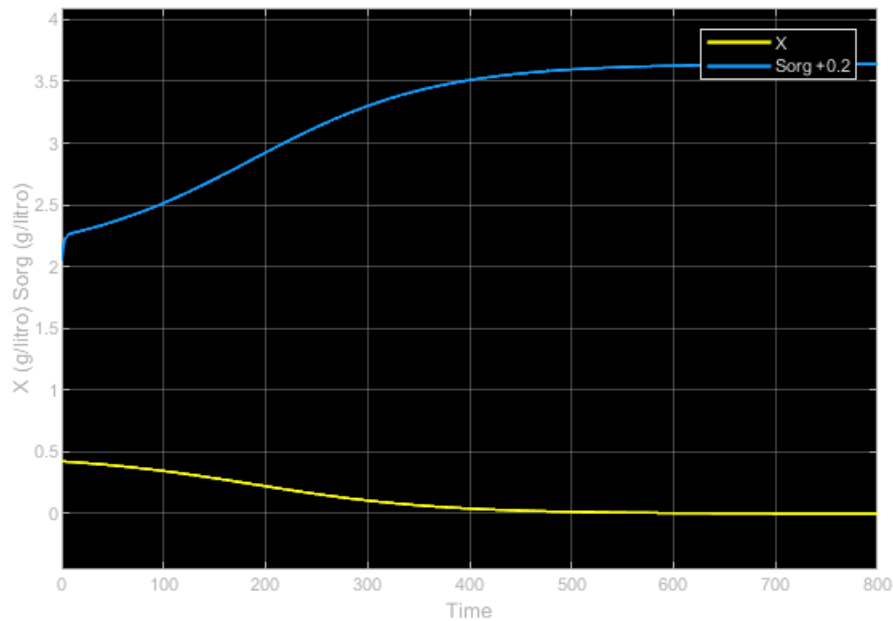


Figure 1: Gráfica de las salidas con variación +0.2 en la entrada Sorgo

Esta grafica en las salidas se obtuvo configurando el escalón de la entrada de Sorg aplicándole 0.2 unidades elevando el valor de la misma a 3.7 g/litro como se muestra en la figura 2.

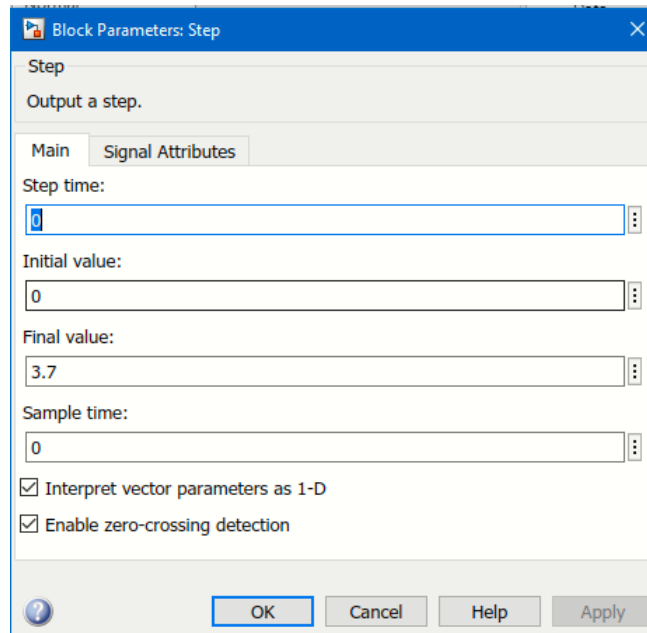


Figure 2: Configuración de la entrada Sorgo con variación de +0.2

Seguidamente se procedió a simular el sistema con la variación de -0.2 g/litro en la entrada Sorgo, las salidas cambiaron como lo muestra ahora la figura 3

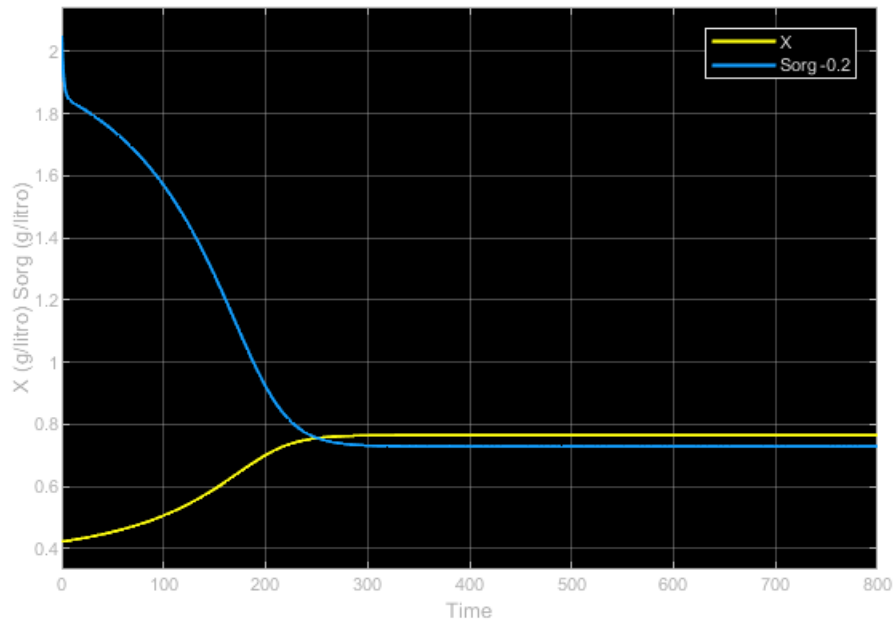


Figure 3: Gráfica de las salidas con variación -0.2 en la entrada Sorgo

Esta se llevo a cabo haciendo una variación en el impulso a la entrada Sorg de -0.2 g/litro como lo identifica la figura 4

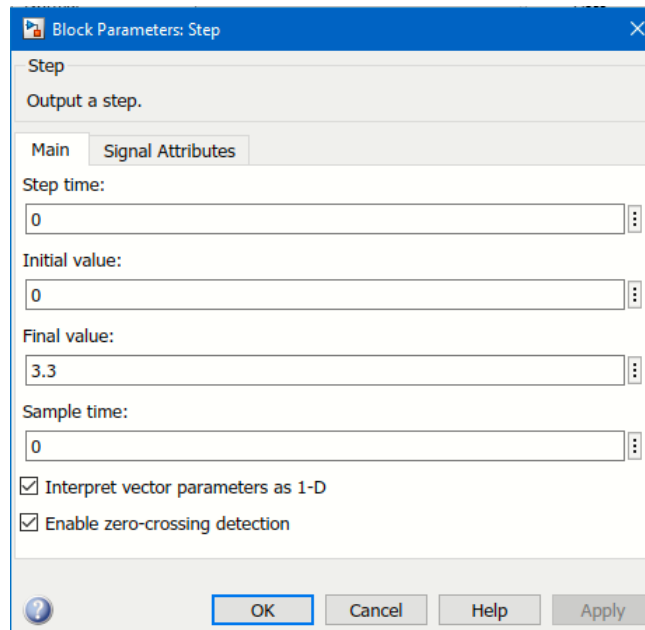


Figure 4: Configuración de la entrada Sorg con variación de -0.2

Si se observan los valores de las dos gráficas obtenidas contra la gráfica numero 6 del articulo en cuestión (figura 5) se puede ver que las gráficas obtenidas se apegan mucho a la del articulo; las amplitudes son las mismas, la variación de las mismas en el eje del tiempo es lo único que varia un poco. Esto puede ser debido a que en el estudio original se incluyo alguna variación que fue omitida para propósitos de este examen.

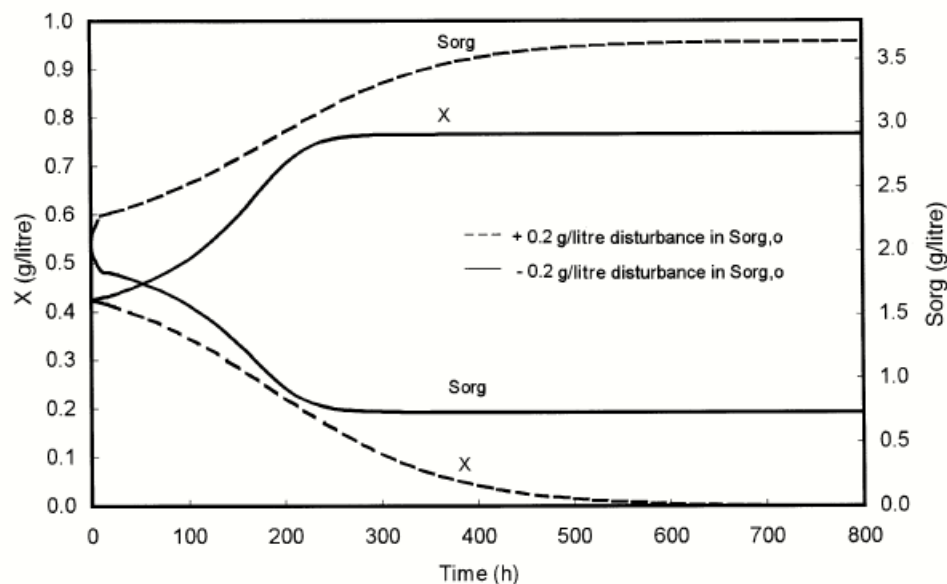


Figure 5: Gráfica de las salidas. Figura 6 del artículo

3 Código de la función S-FUNCTION

El modelo S-FUNCTION (System Function) es ampliamente utilizado para modelar sistemas que no son lineales, que son complicados y que son difíciles de trabajar por otros métodos.

La manera de llevar a cabo la simulación del modelo en cuestión por medio de la función S-FUNCTION es por medio de un archivo de código.m que será el que va a alimentar a la función principal.slx que es la que tiene el modelo S-FUNCTION.

En el archivo que para este examen se denominó "bioreactor.m" se van a escribir las ecuaciones que definen el cálculo de las salidas, las derivadas de los estados.

El archivo principal que para este examen se denominó "bloques.slx", seguirá un algoritmo que contiene las siguientes funciones:

- Una función que contiene la cantidad de entradas, salidas, variables de estado, parámetros y las características del bloque.
- Una función que inicializa los estados del bloque.
- Otra función que calcula las derivadas.
- Una función que calcula las salidas

A continuación se detallarán los pasos utilizados en el código del archivo "bioreactor.m" y una breve explicación de que hace cada parte del mismo. Y también se dará una pequeña explicación de la función principal "bloques.slx":

En el archivo de código se crea una función principal a la que se le van a dar algunas indicaciones como de cuantas entradas va a tener el sistema, cuantas variables de estado van a haber, la cantidad de salidas. Se indicará la cantidad de parámetros que tendrá el modelo; el tiempo de simulación y muestreo.

También en este archivo se definirán subfunciones con las que el sistema trabajará (inicialización, entradas, salidas, variables de estado). Aquí lo que se busca es indicar a S-Function como utilizar las entradas, salidas, variables de estado previamente ingresadas.

Esto hará que simulink sea consistente a la hora de correr la simulación en el modelo de bloques en la función principal.

3.1 El código - "bioreactor.m"

Se debe comenzar creando la función principal, en este caso se denominó 'bioreactor'

```
function bioreactor(block)
```

Función setup, para definir las características básicas del bloque de la s-function

```
setup(block);
```

```
function setup(block)
```

A continuación se van a definir la cantidad de puertos de entrada y salida

```
block.NumInputPorts = 3;  
block.NumOutputPorts = 2;
```

El puerto de setup properties, para que las propiedades ya sean heredadas o dinámicas; en este caso se eligieron propiedades dinámicas

```
block.SetPreCompInpPortInfoToDynamic;  
block.SetPreCompOutPortInfoToDynamic;
```

PROPIEDADES DE LOS PUERTOS DE ENTRADA

```

block.InputPort(1).Dimensions = 1;
block.InputPort(1).DatatypeID = 0; %double
block.InputPort(1).Complexity = 'Real';
block.InputPort(1).DirectFeedthrough = true;

```

```

block.InputPort(2).Dimensions = 1;
block.InputPort(2).DatatypeID = 0; %double
block.InputPort(2).Complexity = 'Real';
block.InputPort(2).DirectFeedthrough = true;

```

```

block.InputPort(3).Dimensions = 1;
block.InputPort(3).DatatypeID = 0; %double
block.InputPort(3).Complexity = 'Real';
block.InputPort(3).DirectFeedthrough = true;

```

PROPIEDADES DE LOS PUERTOS DE SALIDA

```

block.OutputPort(1).Dimensions = 1;
block.OutputPort(1).DatatypeID = 0; %double
block.OutputPort(1).Complexity = 'Real';
block.OutputPort(1).SamplingMode = 'Sample';

```

```

block.OutputPort(2).Dimensions = 1;
block.OutputPort(2).DatatypeID = 0; %double
block.OutputPort(2).Complexity = 'Real';
block.OutputPort(2).SamplingMode = 'Sample';

```

Numero de parámetros. Para este caso se hará un vector de 4 entradas

```

block.NumDialogPrms = 1;

```

Tiempo de muestreo dinámico.

```

block.SampleTimes = [0,0];

```

cantidad de variables de estado. Se eligieron 4 para este caso

```

block.NumContStates = 4;

```

```

block.SimStateCompliance = 'DefaultSimState';

```

A CONTINUACIÓN SE REGISTRAN LOS MÉTODOS INTERNOS DE LA S-FUNCTION

```

block.RegBlockMethod('InitializeConditions', @Inicializacion);
block.RegBlockMethod('Outputs', @Salidas);
block.RegBlockMethod('Derivatives', @ModeloEstados);
block.RegBlockMethod('SetInputPortSamplingMode', @SetInputPortSamplingMode);

```

A CONTINUACIÓN SE DETALLAN CADA UNO DE LOS MÉTODOS INTERNOS DE LA S-FUNCTION

```

function Inicializacion(block)

```

```

block.ContStates.Data = block.DialogPrm(1).Data;

```

Acá se escriben las ecuaciones de salida

```

function Salidas(block)

```

```

x = block.ContStates.Data;
block.OutputPort(1).Data = x(1);
block.OutputPort(2).Data = x(3);

```

Acá se escribe la función que calcula las derivadas de las variables de estado

```
function ModeloEstados(block)
```

A continuación se definen parámetros que identifican las variables de estado

```
sorgo = block.InputPort(1).Data;  
daq = block.InputPort(2).Data;  
ds = block.InputPort(3).Data;  
x = block.ContStates.Data;  
kd = 0.001;  
ke = 0.57;  
ki = 0.470;  
kla = 250;  
klao2 = 43;  
ko = 0.000048;  
ks = 0.001;  
vaq = 1.0;  
vorg = 0.5;  
yo2x = 1/0.338;  
yxs = 0.52;  
umax = 0.534;  
co2 = 0.0373;  
u = ((umax*(x(2)))/(ks+(x(2))+((x(2))^2)/ki))*((x(4))/(ko+(x(4))));  
dphenol = 1.215*((9.75*(exp(-1.8182*x(3)))) - (48.75*(exp(-6.6667*(x(3))))) + 39.0);
```

Este es el valor del estado actual

```
x = block.ContStates.Data;
```

El cálculo de las derivadas de las variables de estado

```
dx1dt = u*(x(1)) - kd*(x(1)) - ke*u*(x(1)) - daq*(x(1));  
dx2dt = kla*(((x(3))/dphenol)-(x(2))) - ((u*(x(1)))/yxs) - daq*(x(2));  
dx3dt = ds*sorgo - kla*(((x(3))/dphenol)-(x(2)))*(vaq/vorg)) - (ds*(x(3)));  
dx4dt = daq*(x(4)) + klao2*(co2-(x(4))) - yo2x*u*(x(1)) - daq*(x(4));
```

Actualización del bloque de la S-function

```
block.Derivatives.Data = [dx1dt;dx2dt;dx3dt;dx4dt];
```

```
function SetInputPortSamplingMode(s, port, mode)  
s.InputPort(port).SamplingMode = mode;  
%end SetInputPortSamplingMode
```

3.2 La función principal

La función principal es el modelo de simulink que emplea el bloque S-Function. A este bloque S-Function se le incorpora el modelo de código para que los bloques de las entradas y las salidas que se le incorporaran hagan lo que deberían de hacer.

La figura 6 presenta el modelo elaborado

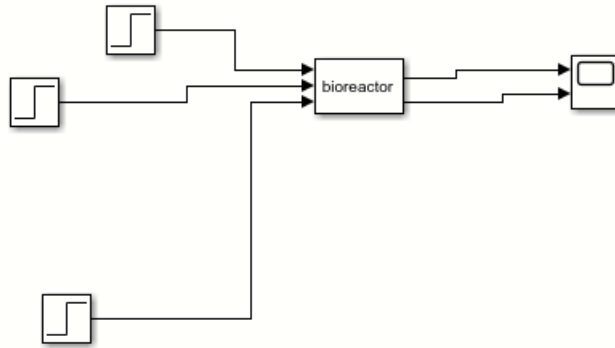


Figure 6: Modelo creado en Simulink

El bloque S-Function se construyo con los parámetros iniciales de las funciones de estado (fig 7)

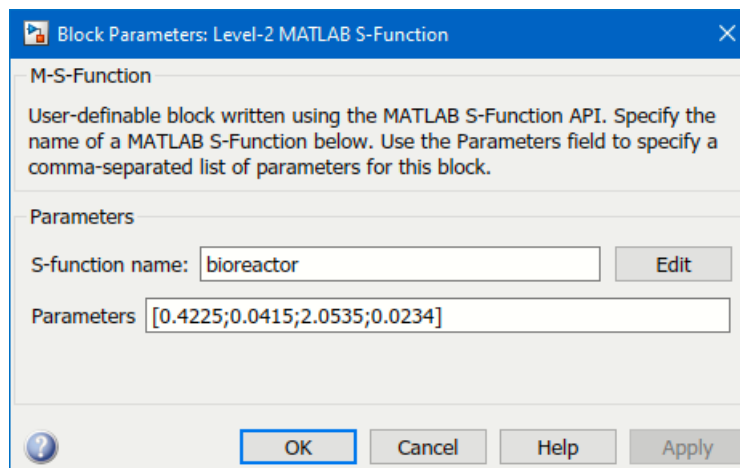
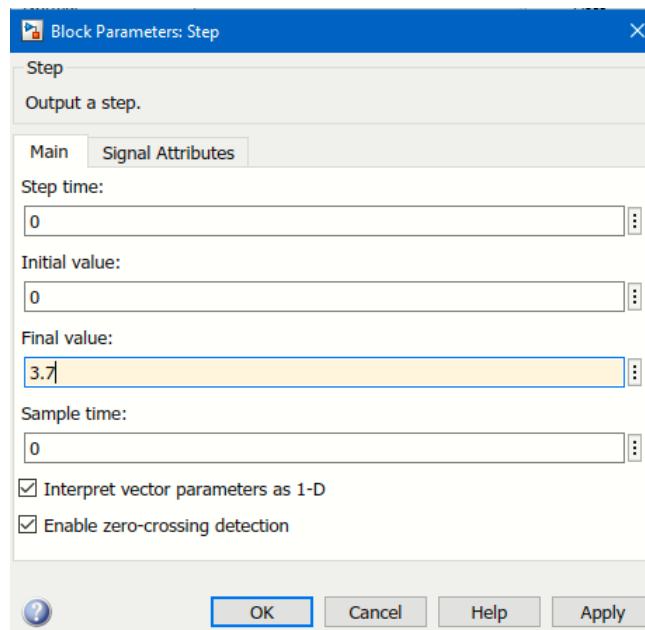


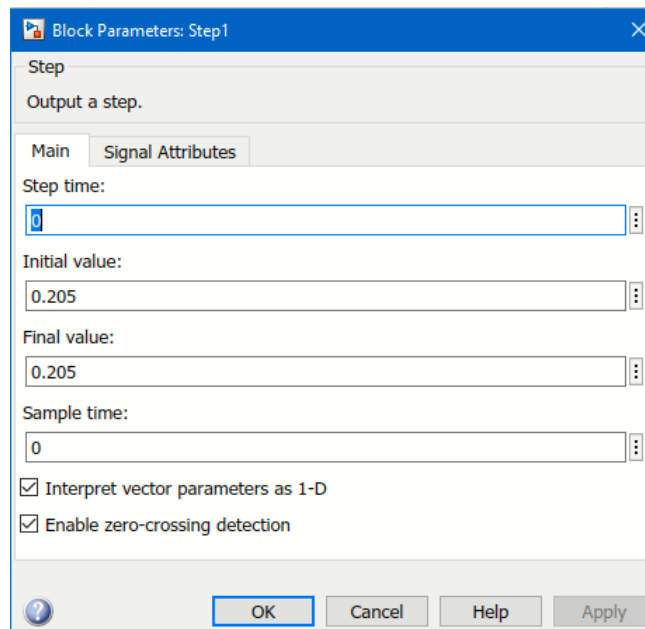
Figure 7: Bloque de S-Function del bioreactor con los parámetros iniciales

Las entradas Sorgo, Daq y Ds se modelaron con bloques de escalón. Las figuras 8, 9 y 10 muestran las entradas ingresadas respectivamente.



The screenshot shows the 'Block Parameters: Step' dialog box. The 'Main' tab is selected. The 'Step time' field is set to 0. The 'Initial value' field is set to 0. The 'Final value' field is set to 3.7. The 'Sample time' field is set to 0. The 'Interpret vector parameters as 1-D' and 'Enable zero-crossing detection' checkboxes are both checked. The 'OK' button is highlighted.

Figure 8: Entrada Sorgo con variacion +2



The screenshot shows the 'Block Parameters: Step1' dialog box. The 'Main' tab is selected. The 'Step time' field is set to 0. The 'Initial value' field is set to 0.205. The 'Final value' field is set to 0.205. The 'Sample time' field is set to 0. The 'Interpret vector parameters as 1-D' and 'Enable zero-crossing detection' checkboxes are both checked. The 'OK' button is highlighted.

Figure 9: Entrada Daq

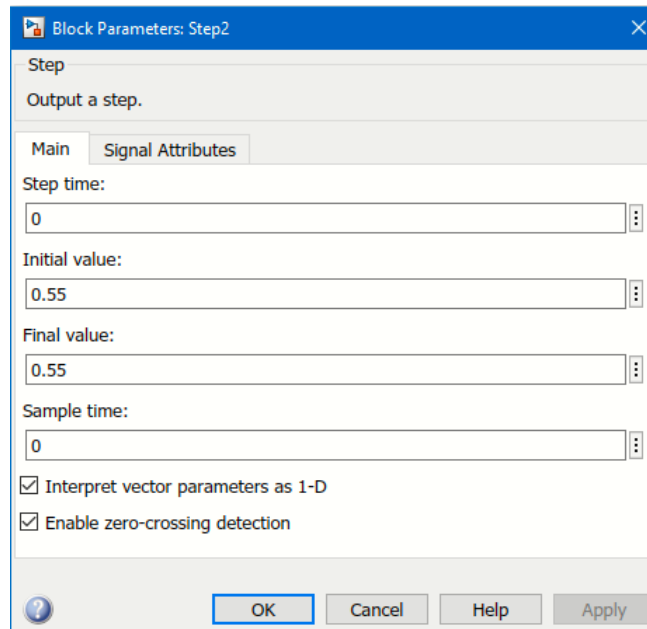


Figure 10: Entrada Ds

Nota: Despues de correr la simulacion se vario la entrada de Sorgo a 3.3 para obtener la otra simulacion en las salidas. La figura 11 muestra el cambio en la entrada

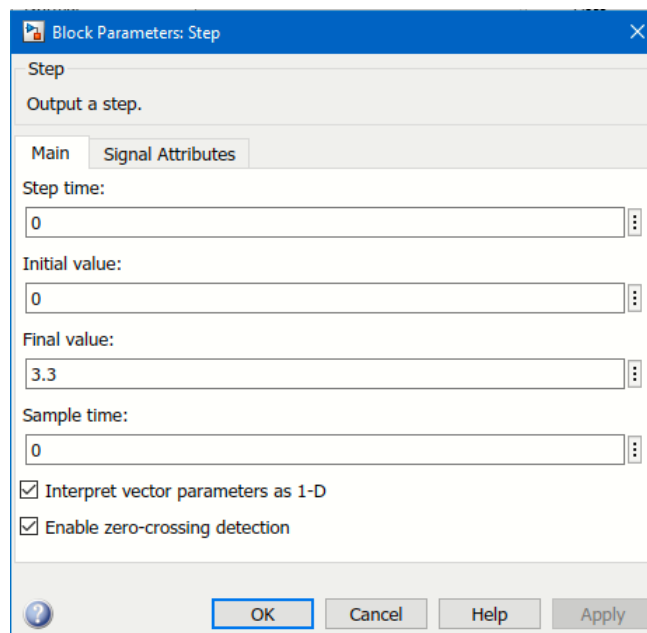


Figure 11: Entrada Sorgo con variacion -0.2

Y para visualizar las salidas se agrego un scope.

4 Modelo ODE45

Este modelo es basado en código de Matlab.

Para modelar el sistema se hizo uso de dos diferentes archivos.m.

Un archivo es el que contiene las ecuaciones de los estados y otras subfunciones que utilizan los estados, estas son Dphenol, μ y también la entrada Sorg.

El otro archivo es la función principal que contiene a la función ODE45.

ODE45 se puede utilizar para simular sistemas en MVE. A la función ODE45 se le debe de ingresar la ecuación diferencial en la forma de MVE (odefun), el tiempo que se quiere simular (tspan) y el estado inicial (x0). Odefun es una función que tenemos que definir de la siguiente manera: dxdt = odefun(t,x), y lo que devuelve esta función es el valor de las derivadas de los estados, y sus entradas son el tiempo y el valor del estado actual. La función odefun es la que le va a pasar todas las ecuaciones de estados, entradas y demás a ODE45 para que corra la simulación pertinente. Finalmente se grafican las salidas deseadas. Para esta prueba al archivo de las funciones de estado se le denomino "variablesdeestado.m" y a la función principal se le denomino "archivoprincipal.m"

4.1 El código de "variablesdeestado.m"

Aquí se implementa la función de los estados del sistema.

```
function dxdt = variablesdeestado(tiempo,x,in)
```

Formulamos el tipo de entrada que después sera llamada por la función principal

```
sorgo=in(1)*heaviside(tiempo);
```

Se definen los parámetros del modelo

```
kd = 0.001;  
ke = 0.57;  
ki = 0.470;  
kla = 250;  
klao2 = 43;  
ko = 0.000048;  
ks = 0.001;  
vaq = 1.0;  
vorg = 0.5;  
yo2x = 1/0.338;  
yxs = 0.52;  
umax = 0.534;  
co2 = 0.0373;
```

Se agregan otras subfunciones necesarias para los estados

```
dphenol = 1.215*((9.75*(exp(-1.8182*(x(3)))) - (48.75*(exp(-6.6667*(x(3)))))+39.0);  
u = ((umax*(x(2)))/(ks+(x(2))+((x(2))^2/ki)))*((x(4))/(ko+(x(4))));
```

Ecuaciones de los estados derivados

```
dxdt = u*(x(1)) - kd*(x(1)) - ke*u*(x(1)) - 0.205*(x(1));  
dsaqdt = kla*(((x(3))/dphenol)-(x(2))) - ((u*(x(1)))/yxs) - 0.205*(x(2));  
dsorgdt = 0.55*sorgo - kla*(((x(3))/dphenol)-(x(2)))*(vaq/vorg) - (0.55*(x(3)));  
dco2dt = 0.205*(x(4)) + klao2*(co2-(x(4))) - yo2x*u*(x(1)) - 0.205*(x(4));
```

Se retorna el valor de los estados derivados para ser utilizados por la función principal

```
dxdt=[dxdt;dsaqdt;dsorgdt;dco2dt];  
end
```

4.2 El código de "archivoprincipal.m"

Se define el tiempo de simulación

```
tiempo = 0:0.01:800;
```

Condiciones iniciales de los estados

```
x0 = [0.4225;0.0415;2.0535;0.0234];
```

Entrada Sorg con perturbación de +0.2

```
amplitud1=3.7;
```

Entrada Sorg con perturbación de -0.2

```
amplitud2=3.3;
```

A continuación se definen las entradas

```
entrada1= [amplitud1];
```

```
entrada2= [amplitud2];
```

ETAPA DE SIMULACIÓN

A continuación se simulan las variables de estado con la variación de +2 en la entrada Sorg

```
[t,x] = ode45(@(tiempo,x) variablesdeestado(tiempo,x,entrada1),tiempo,x0);
```

Acá se simulan las variables de estado con la variación de -2 en la entrada Sorg

```
[t2,x2] = ode45(@(tiempo,x) variablesdeestado(tiempo,x,entrada2),tiempo,x0);
```

A continuación se definen las salidas

Salidas con la variación +2

```
y=x(:,1);
```

```
sorg=x(:,3);
```

Salidas con la variación -2

```
y2=x2(:,1);
```

```
sorg2=x2(:,3);
```

ETAPA DE GRAFICADO

Se grafica para las salidas cuando se le aplico la variación de +2 a la entrada Sorg

```
subplot(2,1,1)
```

```
plot (t,y,'r',t,sorg,'b','Linewidth',1.5)
```

```
grid on
```

```
legend("X","Sorg 3.5+0.2");
```

```
title("Salidas");
```

```
xlabel("Tiempo (h)");
```

```
ylabel("X(g/litro) Sorg(g/litro)");
```

Se grafica para las salidas cuando se le aplico la variación de -2 a la entrada Sorg

```
subplot(2,1,2)
```

```
plot (t,y2,'m',t,sorg2,'g','Linewidth',1.5)
```

```
grid on
```

```
legend("X","Sorg 3.5-0.2");
```

```
title("Salidas");
```

```
xlabel("Tiempo (h)");
```

```
ylabel("X(g/litro) Sorg(g/litro)");
```

A continuación se presentan las gráficas de la simulación. En este caso en contraposición con el método S-Function solo se tuvo que simular una vez y se obtuvieron las dos gráficas al mismo tiempo por la manera en la que se edito el código. La figura 12 muestra las gráficas obtenidas. Se puede notar la similitud a la grafica numero seis del articulo en cuestión (5); las amplitudes son las mismas pero se puede ver que el momento en el tiempo en el que estas varian difiere al de 5, sin embargo estas gráficas son iguales a las gráficas obtenidas mediante el modelo "S-Function" lo que lleva a pensar que una vez mas que es muy probable que se haya omitido algún detalle en el enunciado que no fue de importancia para la realización de este examen simulado, o que los autores del articulo hayan omitido aportar algún dato extra en el articulo que tal vez consideraran irrelevante.

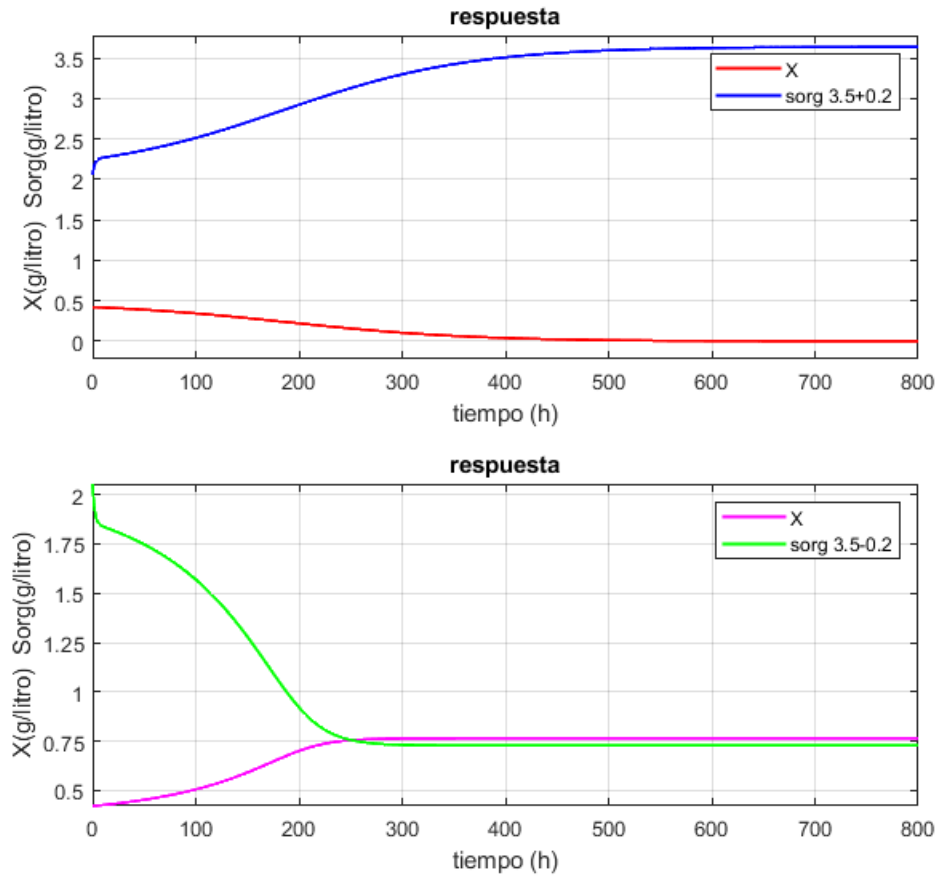


Figure 12: Salidas del modelo con ambas variaciones en la entrada Sorgo.